```
/* Program for Binary Search Tree creation, deletion and traversal */

#include <stdio.h>
#include <stdlib.h>
struct node
{
int ele;
struct node *left;
struct node *right;
};
struct node *head=NULL,*pnew=NULL,*t2,*t1,*t;
void insert_main();
void insert_appro(struct node *t);
void delet_main();
void search1(struct node *t,int n);
void inorder(struct node *t);
int flag;
void main()
{
char ch='y';
int j;
while(ch=='y')
{
printf("1.insert\n2=delete\n3=inorder\n4=exit");
printf("enter your choice");
scanf("%d",&j);
switch(j)
{
case 1:
insert_main();
break;
case 2:
delet_main();
break;
case 3:
inorder(head);
break;
case 4:
return;
break;
}
printf("continue y/n?");
scanf(" %c",&ch);
}
}
//to insert a new element in tree
void insert_main()
{
int n;
printf("enter the data");
scanf("%d",&n);
pnew=(struct node *)malloc(sizeof (struct node));
```

```
pnew->ele=n;
pnew->left=NULL;
pnew->right=NULL;
if(head==NULL)
{
head=pnew;
}
else
insert_appro(head);
}
//function to search for appropriate place to insert and element
void insert_appro(struct node *t)
{
if((pnew->ele>t->ele)&&(t->right!=NULL))
insert_appro(t->right);
else if ((pnew->ele>t->ele)&&(t->right==NULL))
t->right=pnew;
else if((pnew->ele<t->ele)&&(t->left!=NULL))
insert_appro(t->left);
else if((pnew->ele<t->ele)&&(t->left==NULL))
t->left=pnew;
}
//function for deleting nodes at different places
void delet(struct node *t)
{
int k;
//to delete a leaf node
if((t->left==NULL)&&(t->right==NULL))
{
if(t1->left==t)
{
t1->left=NULL;
}
else
{
t1->right=NULL;
}
t=NULL;
free(t);
return;
}
//to delete left hand node
else if(t->right==NULL)
{
if(t1==t)
{
head=t->left;
t1=head;
}
else if(t1->left==t)
{
t1->left=t->left;
}
```

```
else
{
t1->right=t->left;
}
t=NULL;
free(t);
return;
}
//to delete node at right hand child
else if(t->left==NULL)
{
if(t1==t)
{
head=t->right;
t1=head;
}
else if(t1->right==t)
t1->right=t->right;
else
t1->left=t->right;
t=NULL;
free(t);
return;
}
// to delete nodes with both left and right hand childs
else if((t->left!=NULL)&&(t->right!=NULL))
{
t2=head;
if(t->right!=NULL)
{
k=small(t->right);
flag=1;
}
else
{
k=large(t->left);
flag=2;
}
search1(head,k);
t->ele=k;
}
}
//function to search for smallest element in the tree
int small(struct node *t)
{
t2=t;
if(t->left!=NULL)
{
t2=t;
return(small(t->left));
}
else
return(t->ele);
```

```
}
//function to search for largest element in the tree
int large(struct node *t)
{
if(t->right!=NULL)
{
t2=t;
return(large(t->right));
}
else
return(t->ele);
}

void search1(struct node *t, int n)
{
if((n>t->ele))
{
t1=t;
search1(t->right,n);
}
else if((n<t->ele))
{
t1=t;
search1(t->left,n);
}
else if((n==t->ele))
{
delet(t);
}
}
//function for main deletion
void delet_main()
{
int n;
if(head==NULL)
{
printf("----------------------/n");
printf("no elements");
return;
printf("enter the data to delete");
scanf("%d",&n);
t1=head;
t2=head;
search1(head,n);
}}
// traversal used for display purpose
void inorder(struct node *t)
{
if(head==NULL){
printf("------------------------\n");
printf("no elements");
return;
}
```

```
if(t->left!=NULL)
inorder(t->left);
printf("%d-->",t->ele);
if(t->right!=NULL)
inorder(t->right);
}
```